Andrew Laird's 256K RAM Expansion

Contents

Introduction

This project will take a VZ200 or VZ300 to the maximum 34K user RAM,and allow an additional 224K in 14 x 16K switched banks. The design is based on Joe Leon's 128K Sideways RAM project as described in the HVVZUG Journal, but is a little simpler due to the ready availablility of large static RAM IC's. I also consulted Lloyd Butler's article in Amateur Radio April 1988.

The RAM expansion occupies from 8000H to FFFFH (as does the original DSE 64K RAM expansion). This is achieved by enabling the static RAM with A15 (corresponding to 8000H). Because the chip select is negative edge triggered, A15 is inverted through IC 2 (LSHCT04 - hex inverter). The top 16K is switched as per a value written to the memory bank I/O area (addresses 70H-7FH - as the number of banks is small just the top byte of the address is used in practice). For the purpose of this project, the relevant banks are 0 through 15, and hence the relevant data lines are D0 through D3. As we only need to address one RAM IC, the design is fairly simple because address lines A14 through A17 simply follow the data lines (sort of - the complicated bit is addressed below).

The current bank selected is stored via the latches in IC 4 (LS74175), which is cleared via the -RESET line when the VZ-200 is turned on. The four latches correspond to data lines 0 through 3. The latch is clocked when a value is written to I/O address 70H, or 112 decimal (as the bottom half of the address is ignored, in practice you can write the value to any I/O address between 70H and 7FH). The clock pulse is generated by IC1 (74HCT138 - a 3 to 8 multiplexer) which uses the four address lines A4 through A7, and the -IORQ and -WR lines as inputs.

Note, however, that the biggest problem in attempting the design is in maintaining compatibility with the original 64K RAM expansion. With the original cartridge bank 0 was found at 8000H to BFFFH, while bank 1 was at C000H to FFFFH, with the other banks switched in to C000H to 7FFFH as selected (by writing 2 or 3 to I/O address 112). This means that in effect bank 0 and bank 1 are the same setting. This is achieved in the new design by ORing the stored D0 with inverted result of ANDing the other three stored data lines together. (In the design the OR function is arrived at through spare gates in the 74HCT04 and 74LS11 to cut down on the chip count - OR is equivalent to inverting the inputs and ANDing them together and inverting the result.)

Note also that this design uses a 512K x 8 bit SRAM (the two sizes available to me were 512K or 128K) but a 256K x 8 bit SRAM will suffice if you can find one. A 684000 was used as this was available at the right price.

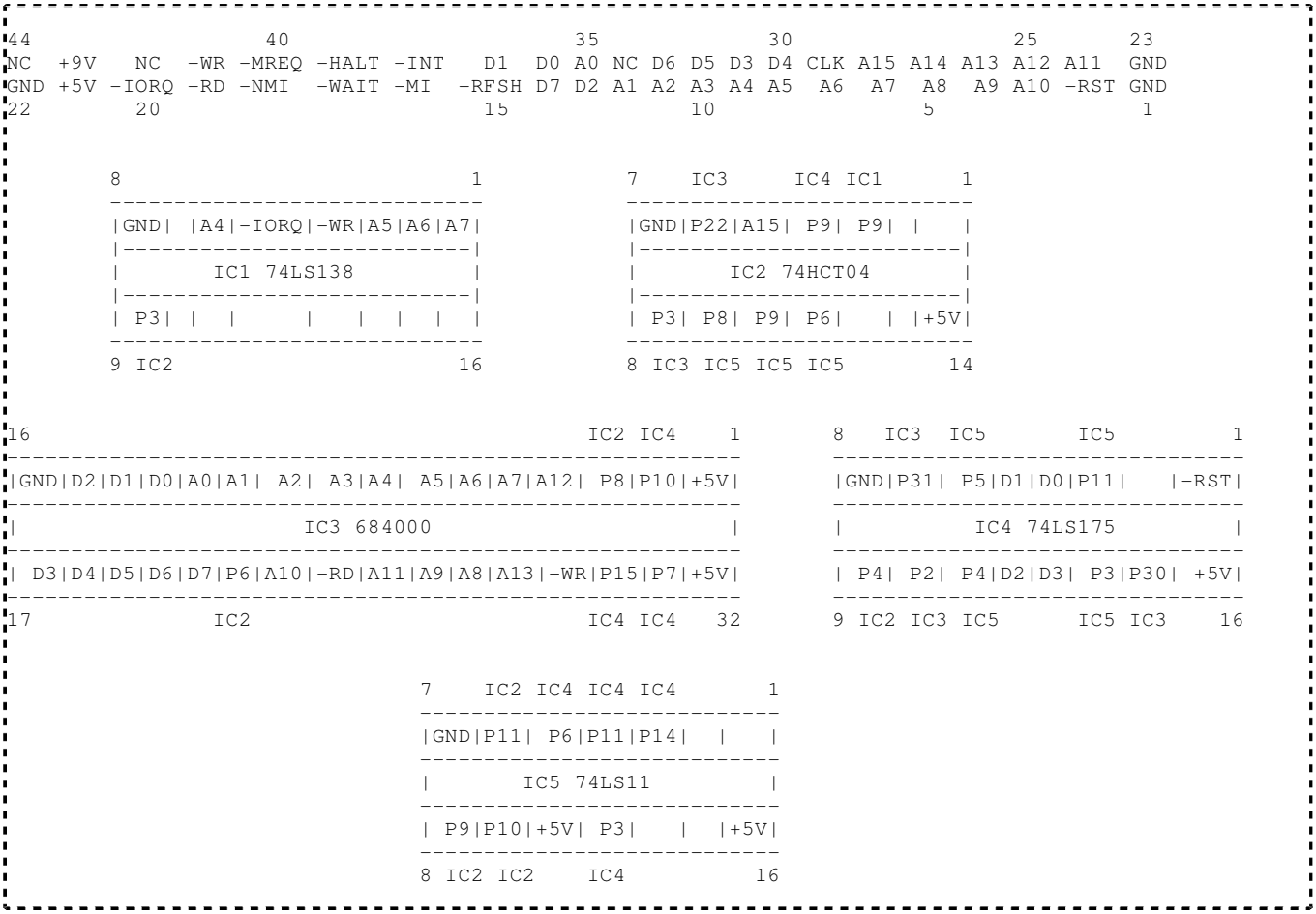
NOTE: If you really wanted to, you could have a 64MB RAM expansion, as there are 16 bank switch I/O address

available for use, and each address is a byte wide, which allows for 256 banks per I/O address. (16 x 256 x 16K = 64MB) I imagine it would be prohibitively expensive if static RAM was used, though, and would require extensive addressing circuitry.

Design

The following is the design I used. In order to build it all you need to know is which wires to connect, but it is worth spending a little time trying to understand how it works (datasheets for the ICs can be downloaded from datasheetarchive.com (http://datasheetarchive.com/)). That way you can design your own.

The connections all refer to the edge connector unless otherwise noted. P stands for pin. A blank means no connection. For example, the designation A15 at pin 5 of IC2 means connect pin 5 of IC2 to A15 on the edge connector, while P3 IC3 at pin 8 means connect pin 8 to pin 3 of IC3. Note that there is a corresponding entry P8 IC2 at pin 3 of IC3. At the top of the diagram is the edge connector pinout as seen from the top side of the board (this is upside down compared to the VZ200 expansion port due to the right angle mount ie the top row on the expansion port is mapped to the lower row on the board).



Parts List

- 44 way edge connector, preferably right angle mounting - I salvaged one from a VZ300 16K RAM expansion. This was pretty difficult and I broke 3 legs while doing this. Fortunately, 2 were not needed and I managed to solder some wirewrap wire to the third, which worked.
- suitable prototype board - I used a 120mm x 80mm board with individual copper per hole. This board has 46 holes at the top and suited the edge connector.
- static RAM, at least 256K x 8bit - as a 256K RAM wasn't available to me I chose to use a 512K RAM instead. This was pretty cheap and I figured I could always use the other 256K or so for future expansion if I could be bothered.

- 1 each 74LS138, 74LS04, 74LS175, 74LS11 (or their HCT equivalents)
- IC sockets to suit all the IC's

Approximate cost $25

Construction

This was the first project I had attempted for a number of years, but I had made numerous kits as a teenager, so I was fairly confident I could get this built right the first time. I decided to run the wires on the top of the board, as this allowed me to used the stripped ends of the wire to reach from the hole next to the IC socket to the leg of the socket itself. This mostly worked well, but sometimes I ended up with too much solder and bridges. These were easily fixed. Occasionally I had to duplicate lines and these were run from the next hole out again from the IC socket already attached to the relevant line. This was easier than trying to connect multiple lines to the edge connector where space was more of an issue.

To supply the IC's with Vcc and GND, I ran a short +5V rail down the left hand side of the board, with a corresponding GND rail down the right hand side of the board. The rails were made from stripped wire-wrap and the normal wire soldered on to that.

I managed to find some rainbow ribbon cable in my junk box, so I used that to run most of the lines from the 684000. This was very helpful when trying to work out which line to run where.

As it was, I ended up with a bit of a bird's nest of wire but I had allowed enough slack to get all the IC's in to their sockets without too much hassle. If I did it again I would allow enough length to route the wire around the IC sockets rather than over them, as this would be tidier.

Along the way I checked connections before and after soldering, and occasionally got the multimeter out to ensure that no bridging was occuring between pins. The checking was handy as I picked up a couple of mistakes along the way.

Testing

It was with some trepidation that I plugged the unit in to my one and only VZ200 and turned the power on.

The normal screen came up and I was relieved.

I checked the top of memory pointer (30897,30898) and both locations read 255. I was very happy!

I did some quick bank switching testing by hand and this seemed to work, so I went on to further tests. Looking through my small assortment of tapes I decided to load Hoppy, a 24K program. The tape loaded fine, the instructions and intro screen worked, but when I went to play the game it crashed badly. Disaster!

Anyway, I decided to test the RAM with a simple BASIC program. After some messing around with a BASIC program I realised that my POKEs were interfering with the operation of the BASIC program itself, so I went back to Lloyd Butler's article and realised that he had set the stack pointer and top of memory pointer to below his RAM expansion and then tested the RAM, so I decided to do the same. Note that my RAM expansion starts at 8000H or 32768 which is equivalent to -32768, and overlaps some of the VZ200 RAM.

When I checked the stack pointer at power up it was about 50 below top of memory so I have used a figure of 55 below in the following examples.

This allows us to check the first 32K of the RAM expansion

```
POKE30880,200:POKE30881,127 (set stack pointer to 32712)<br>
POKE30897,255:POKE30898,127 (set top of memory to 32767)<br>

10 CLS
20 FORI=-32768TO-1:PRINT@0,I; (loop through the memory and print current memory location)
30 POKEI,0:IFPEEK(I)<>0THENPRINT@128,"ERROR: ";I:END (write 0's and test)
40 POKEI,255:IFPEEK(I)<>255THENPRINT@128,"ERROR: ";I:END (write 1's and test)
50 NEXT
```

Very simple, but reasonably fast (completed in about 22 minutes) and a good test, plus it has visual feedback on the progress.

After this test didn't flag any errors I decided that the Hoppy tape was quite possibly corrupted. I loaded Star Blaster (another program requiring 24K) and this worked fine.

Note that in normal circumstances trying to run a BASIC program with bank switching won't work, as the stack and possibly other important stuff will get switched out as well (which will cause the VZ to reset). The good news is that by setting the top of memory and stack pointers as above, you can use bank switching in BASIC.

Here's a modified listing to test that bank switching is working as expected.

This allows us to check the top 16K of the RAM expansion

```
POKE30880,200:POKE30881,191 (set stack pointer to 49096)
POKE30897,255:POKE30898,191 (set top of memory to 49151)

10 CLS
15 FORJ=1TO15:OUT112,J:PRINT@0,"BANK ";J; (loop through the 15 banks)
20 FORI=-16384TO-1:PRINT@32,I; (loop through the memory and print current memory location)
30 POKEI,0:IFPEEK(I)<>0THENPRINT@128,"ERROR: ";I:END (write 0's and test)
40 POKEI,255:IFPEEK(I)<>255THENPRINT@128,"ERROR: ";I:END (write 1's and test)
45 POKEI,J (write the bank number into the bank for later testing)
50 NEXT:NEXT
60 FORJ=1TO15:OUT112,J:PRINT@0,"BANK ";J;
70 FORI=-16384TO-1:PRINT@32,I;
80 IFPEEK(I)<>JTHENPRINT@128,"ERROR: ";I:END (check that the bank number corresponds to saved value)
90 NEXT:NEXT
```

NB This takes a long time (hours). I let it run overnight (checking a few iterations before going to bed) and came back in the morning to see that my RAM expansion had passed the test.

Possible Uses

As we now know how to do bank switching in BASIC, this raises a lot of possibilities. 15 banks of 16K gives 90 frames of animation (admittedly not that much even if you are running the frames at 10 frames per second or slower), or 90 screens for a slideshow, or 90 different levels for a game, and so on. Of course, writing to RAM is slow in BASIC, so you would probably want a machine code block load subroutine you could call from BASIC to load up a screen quickly. After I have played around a bit I will release some software ideas for the project.

Andrew Laird, April 2005

Back to Complete Hardware Project List

Back to The List of Lists

Back to Andrew Laird

Back to Main Page
